3 STATIC TESTING

Geoff Thompson

INTRODUCTION

This chapter provides an introduction to an important area of software testing – static techniques. Static techniques test software without executing it. They are therefore important because they can find errors and defects before code is executed and therefore earlier in the life cycle of a project, making corrections easier and cheaper to achieve than for the same defects found during test execution. Review techniques are central to the static testing approach, and in this chapter we will look at the alternative types of review and how they fit with the test process that was defined in Chapter 2.

The chapter also considers static analysis of the developed code, which is a technique for examining code without executing it to identify actual and potential problems. With the advances in coding languages and the legacy status of many older systems, static testing is often impossible to achieve manually, so our focus will therefore be on the types of static testing that can be completed using tools. In this chapter we focus on the techniques of static testing; the tools that are referred to are described in Chapter 6.

Learning objectives

The learning objectives for this chapter are listed below. You can confirm that you have achieved these by using the self-assessment questions at the start of the chapter, the 'Check of understanding' boxes distributed throughout the text, and the example examination questions provided at the end of the chapter. The chapter summary will remind you of the key ideas.

The sections are allocated a K number to represent the level of understanding required for that section; where an individual element has a lower K number than the section as a whole this is indicated for that topic; for an explanation of the K numbers see the Introduction.

Reviews and the test process (K2)

- Recognise software work-products that can be examined by the different static techniques. (K1)
- Describe the importance and value of considering static techniques for the assessment of software work-products.

- Explain the difference between static and dynamic techniques, considering objectives, types of defects to be identified, and the role of these techniques within the software life cycle.

Review process (K2)

- Recall the phases, roles and responsibilities of a typical formal review. (K1)
- Explain the differences between different types of review: informal review, technical review, walkthrough and inspection.
- Explain the factors for successful performance of reviews.

Static analysis by tools (K2)

- Recall typical defects and errors identified by static analysis and compare them with reviews and dynamic testing. (K1)
- Describe using examples, the typical benefits of static analysis.
- List typical code and design defects that may be identified by static analysis tools. (K1)

Self-assessment questions

The following questions have been designed to enable you to check your current level of understanding for the topics in this chapter. The answers are at the end of the chapter.

Question SA1 (K1)

One of the roles in a review is that of moderator, which of the following best describes this role?

a. Plans the review, runs the review meeting and ensures that follow-up activities are completed.
b. Allocates time in the plan, decides which reviews will take place and that the benefits are delivered.
c. Writes the document to be reviewed, agrees that the document can be reviewed, and updates the document with any changes.
d. Documents all issues raised in the review meeting, records problems and open points.

Question SA2 (K2)

Which of the following statements are correct for walkthroughs?

(i) Often led by the author.
(ii) Documented and defined results.
(iii) All participants have defined roles.
(iv) Used to aid learning.
(v) Main purpose is to find defects.

a. (i) and (v) are correct.
b. (ii) and (iii) are correct.
c. (i) and (iv) are correct.
d. (iii) and (iv) are correct.

Question SA3 (K1)
What do static analysis tools analyse?

a. Design
b. Test cases
c. Requirements
d. Program code

BACKGROUND TO STATIC TECHNIQUES

Static testing techniques are those techniques that test software without executing the code. This includes both the testing of work-products other than code, typically requirements or specification documents, and the testing of code without actually executing it. The first of these is known as a review and is typically used to find and remove errors and ambiguities in documents before they are used in the development process, thus reducing one source of defects in the code; the second is known as static analysis, and it enables code to be analysed for structural defects or systematic programming weaknesses that may lead to defects.

Reviews are normally completed manually; static analysis is normally completed automatically using tools. The tools used for static analysis will be described in Chapter 6.

REVIEWS AND THE TEST PROCESS

A review is a systematic examination of a document by one or more people with the main aim of finding and removing errors. Giving a draft document to a colleague to read is the simplest example of a review, and one which can usually yield a larger crop of errors than we would have anticipated (see Chapter 5 regarding 'World view' to understand why).

Reviews can be used to test anything that is written or typed; this can include documents such as requirement specifications, system designs, code, test plans and test cases. Reviews represent the first form of testing that can take place during a software development life cycle, since the documents reviewed are normally ready long before the code has been written. The practice of testing specification documents by reviewing them early on in the life cycle helps to identify defects before they become part of the executable code, and so makes those defects cheaper and easier to remove. The same defect, if found during dynamic test execution, would incur the extra cost of initially creating and testing the defective code, diagnosing the source of the defect, correcting the problem and rewriting the code to eliminate the defect. Reviewing code against development standards can also prevent defects from appearing in test execution, though in this case, as the code has already been written, not all the additional costs and delays are avoided.

Important as cost and time saving are, though, there are also other important benefits of finding defects early in the life cycle, among them the following:

- Development productivity can be improved and timescales reduced because the correction of defects in early work-products will help to ensure that those work-products are clear and unambiguous. This should enable a developer to move more quickly through the process of writing code. Also, if defects are removed before they become executable code there will be fewer errors to find and fix during test execution.
- Testing costs and time can be reduced by removing the main delays in test execution, which arise when defects are found after they have become failures and the tester has to wait for a fix to be delivered. By reviewing the code and removing defects before they become failures the tester can move more swiftly through test execution.
- Reductions in lifetime costs can be achieved because fewer defects in the final software ensure that ongoing support costs will be lower.
- Improved communication results as authors and their peers discuss and refine any ambiguous content discovered during review to ensure that all involved understand exactly what is being delivered.

The types of defects most typically found by reviews are:

- Deviations from standards either internally defined and managed or regulatory/legally defined by Parliament or perhaps a trade organisation.
- Requirements defects – for example, the requirements are ambiguous, or there are missing elements.
- Design defects – for example, the design does not match the requirements.
- Insufficient maintainability – for example, the code is too complex to maintain.
- Incorrect interface specifications – for example, the interface specification does not match the design or the receiving or sending interface.

All reviews aim to find defects, but some types of review find certain types of defects more effectively and efficiently than others.

Review process

Review processes can vary widely in their level of formality, where formality relates to the level of structure and documentation associated with the activity. Some types of review are completely informal, while others are very formal. The decision on the appropriate level of formality for a review is usually based on combinations of the following factors:

- The maturity of the development process: the more mature the process is, the more formal reviews tend to be.
- Legal or regulatory requirements. These are used to govern the software development activities in certain industries, notably in safety-critical areas such as railway signalling, and determine what kinds of review should take place.

- The need for an audit trail. Formal review processes ensure that it is possible to trace backwards throughout the software development life cycle. The level of formality in the types of review used can help to raise the level of audit trail.

Reviews can also have a variety of objectives,where the term 'review objective' identifies the main focus for a review. Typical review objectives include:

- Finding defects.
- Gaining understanding.
- Generating discussion.
- Decision making by consensus.

The way a review is conducted will depend on what its specific objective is, so a review aimed primarily at finding defects will be quite different from one that is aimed at gaining understanding of a document.

Basic review process

All reviews, formal and informal alike, exhibit the same basic elements of process:

- The document under review is studied by the reviewers.
- Reviewers identify issues or problems and inform the author either verbally or in a documented form, which might be as formal as raising a defect report or as informal as annotating the document under review.
- The author decides on any action to take in response to the comments and updates the document accordingly.

This basic process is always present, but in the more formal reviews it is elaborated to include additional stages and more attention to documentation and measurement.

Activities of a formal review

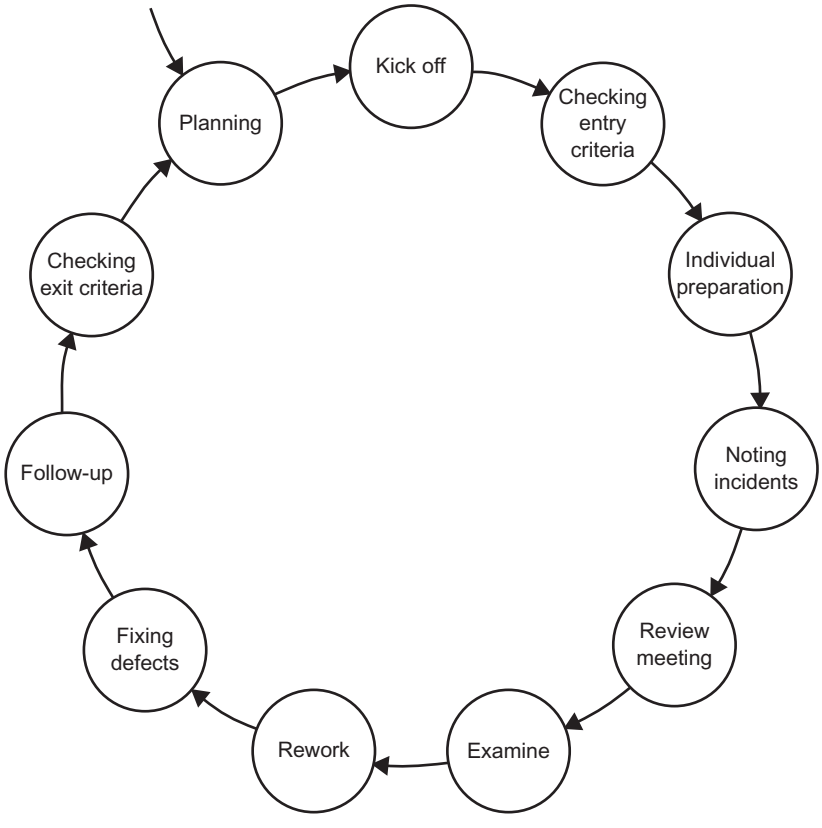
Reviews at the more formal end of the spectrum, such as technical reviews and inspections, share certain characteristics that differentiate them from the less formal reviews, of which walkthroughs are a typical example.

Figure 3.1 shows the key stages that characterise formal reviews.

The following list explains the key stages in more detail:

- Planning:
    - Selecting the personnel – ensuring that those selected can and will add value to the process. There is little point in selecting a reviewer who will agree with everything written by the author without question. As a rule of thumb it is best to include some reviewers who are from a different part of the organisation, who are known to be 'picky', and known to be dissenters.

Figure 3.1 Stages of a formal review



Reviews, like weddings, are enhanced by including 'something old, something new, something borrowed, something blue'. In this case 'something old' would be an experienced practitioner; 'something new' would be a new or inexperienced team member; 'something borrowed' would be someone from a different team; 'something blue' would be the dissenter who is hard to please. At the earliest stage of the process a review leader must be identified. This is the person who will coordinate all of the review activity.

- Allocating roles – each reviewer is given a role to provide them with a unique focus on the document under review. Someone in a tester role might be checking for testability and clarity of definition, while someone in a user role might look for simplicity and a clear relationship to business values. This approach ensures that, although all reviewers are working on the same document, each individual is looking at it from a different perspective.
- Defining the entry and exit criteria, especially for the most formal review types (e.g. inspection).

  - Selecting the parts of documents to be reviewed (not always required; this will depend on the size of the document: a large document may need to be split into smaller parts and each part reviewed by a different person to ensure the whole document is reviewed fully).

- Kick-off: distributing documents; explaining the objectives, process and documents to the participants; and checking entry criteria (for more formal review types such as inspections). This can be run as a meeting or simply by sending out the details to the reviewers. The method used will depend on timescales and the volume of information to pass on. A lot of information can be disseminated better by a meeting rather than expecting reviewers to read pages of text.
- Review entry criteria: this stage is where the entry criteria agreed earlier are checked to ensure that they have been met, so that the review can continue – this is mainly used in the more formal review types such as inspections.
- Individual preparation: work done by each of the participants on their own before the review meeting, which would include reading the source documents, noting potential defects, questions and comments. This is a key task and may actually be time-boxed, e.g. participants may be given two hours to complete the preparation.
- Noting incidents: in this stage the potential defects, questions and comments found during individual preparation are logged.
- Review meeting: this may include discussion regarding any defects found, or simply just a log of defects found. The more formal review types like inspections will have documented results or minutes. The meeting participants may simply note defects for the author to correct; they might also make recommendations for handling or correcting the defects. The approach taken will have been decided at the kick-off stage so that all participants are aware of what is required of them. The decision as to which approach to take may be based on one or all of the following factors:

  - Time available (if time is short the meeting may only collect defects).
  - Requirements of the author (if the author would like help in correcting defects).
  - Type of review (in an inspection only the collection of defects is allowed – there is never any discussion).

- Examine: this includes the recording of the physical meetings or tracking any group electronic communications.
- Rework: after a review meeting the author will have a series of defects to correct; correcting the defects is called rework.
- Fixing defects: here the author will be fixing defects that were found and agreed as requiring a fix.

- Follow-up: the review leader will check that the agreed defects have been addressed and will gather metrics such as how much time was spent on the review and how many defects were found. The review leader will also check the exit criteria (for more formal review types such as inspections) to ensure that they have been met.
- Checking exit criteria: at this stage the exit criteria defined at the start of the process are checked to ensure that all exit criteria have been met so that the review can be officially closed as finished.

Roles and responsibilities

The role of each reviewer is to look at documents belonging to them from their assigned perspective; this may include the use of checklists. For example, a checklist based on a particular perspective (such as user, maintainer, tester or operations) may be used, or a more general checklist (such as typical requirements problems) may be used to identify defects.

In addition to these assigned review roles the review process itself defines specific roles and responsibilities that should be fulfilled for formal reviews. They are:

- Manager: the manager decides on what is to be reviewed (if not already defined), ensures there is sufficient time allocated in the project plan for all of the required review activities, and determines if the review objectives have been met. Managers do not normally get involved in the actual review process unless they can add real value, e.g. they have technical knowledge key to the review.
- Moderator: the moderator is sometimes known as the review leader. This is the person who leads the review of the document or set of documents, including planning the review, running the meeting, and follow-ups after the meeting. If necessary, the moderator may mediate between the various points of view and is often the person upon whom the success of the review rests. The moderator will also make the final decision as to whether to release an updated document.
- Author: The author is the writer or person with chief responsibility for the development of the document(s) to be reviewed. The author will in most instances also take responsibility for fixing any agreed defects.
- Reviewers: These are individuals with a specific technical or business background (also called checkers or inspectors) who, after the necessary preparation, identify and describe findings (e.g. defects) in the product under review. As discussed above, reviewers should be chosen to represent different perspectives and roles in the review process and take part in any review meetings.
- Scribe (or recorder): The scribe attends the review meeting and documents all of the issues and defects, problems and open points that were identified during the meeting.

An additional role not normally associated with reviews is that of the tester. Testers have a particular role to play in relation to document reviews. In their

test analysis role they will be required to analyse a document to enable the development of tests. In analysing the document they will also review it, e.g. in starting to build end-to-end scenarios they will notice if there is a 'hole' in the requirements that will stop the business functioning, such as a process that is missing or some data that is not available at a given point. So effectively a tester can either be formally invited to review a document or may do so by default in carrying out the tester's normal test analysis role.

CHECK OF UNDERSTANDING

(1) Identify three benefits of reviews.

(2) What happens during the planning phase of a review?

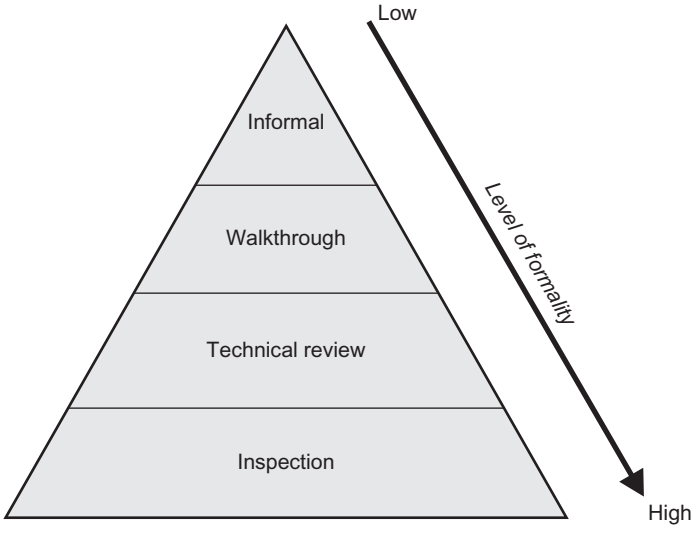
(3) Who manages the review process?

Types of review

A single document may be subject to many different review types: for example, an informal review may be carried out before the document is subjected to a technical review or, depending on the level of risk, a technical review or inspection may take place before a walkthrough with a customer.

Figure 3.2 shows the different levels of formality by review type.

Figure 3.2 Formality of reviews



Each type of review has its own defining characteristics. We identify four review types to cover the spectrum of formality. These are usually known as:

(1) Informal review (least formal). Key characteristics:

- There is no formal process underpinning the review.
- The review may be documented but this is not required; many informal reviews are not documented.
- There may be some variations in the usefulness of the review depending on the reviewer, e.g. the reviewer does not have the technical skills but is just available to check quickly and ensure that the document makes sense.
- The main purpose is to find defects and this is an inexpensive way to achieve some limited benefit.
- The review may be implemented by pair programming (where one programmer reviews the code of the other 'pair programmer') or by a technical lead reviewing designs and code.

(2) Walkthrough. Key characteristics:

- The meeting is led by the author of the document under review and attended by members of the author's peer group.
- Review sessions are open-ended and may vary in practice from quite informal to very formal.
- Preparation by reviewers before the walkthrough meeting, production of a review report or a list of findings, and appointment of a scribe who is not the author are all optional components that are sometimes present.
- The main purposes are to enable learning about the content of the document under review, to help team members gain an understanding of the content of the document, and to find defects.
- Walkthroughs typically explore scenarios, or conduct dry runs of code or process.

(3) Technical review. Key characteristics:

- Technical reviews are documented and use a well-defined defect detection process that includes peers and technical experts.
- The review is usually performed as a peer review without management participation and is ideally led by a trained moderator who is not the author.
- Reviewers prepare for the review meeting, optionally using checklists, and prepare a review report with a list of findings.
- Technical reviews may vary in practice from the quite informal to very formal and have a number of purposes, including: discussion, decision making, evaluation of alternatives, finding defects, solving technical problems and checking conformance to specifications and standards.

(4) Inspection (most formal). Key characteristics:

- Inspections are led by a trained moderator who is not the author and usually involve peer examination of a document; individual inspectors work within defined roles.
- The inspection process is formal, based on rules and checklists, and uses entry and exit criteria.
- Pre-meeting preparation is essential, which would include reading of any source documents to ensure consistency.
- An inspection report, with a list of findings, is produced, which includes metrics that can be used to aid improvements to the process as well as correcting defects in the document under review.
- After the meeting a formal follow-up process is used to ensure that corrective action is completed and timely.
- The main purpose is to find defects, and process improvement may be a secondary purpose.

In reality the lines between the review types often get blurred and what is seen as a technical review in one company may be seen as an inspection in another. The above is the 'classic view' of reviews. The key for each company is to agree the objectives and benefits of the reviews that they plan to carry out.

Success factors for reviews
When measuring the success of a particular review the following suggested success factors should be considered:

- Each review should have a clearly predefined and agreed objective and the right people should be involved to ensure the objective is met. For example, in an inspection each reviewer will have a defined role and therefore needs the experience to fulfil that role; this should include testers as valued reviewers.
- Any defects found are welcomed, and expressed objectively.
- The review should be seen as being conducted within an atmosphere of trust, so that the outcome will not be used for the evaluation of the participants, and that the people issues and psychological aspects are dealt with (e.g. making it a positive experience for the author and all participants).
- Review techniques (both formal and informal) that are suitable to the type and level of software work-products and reviewers (this is especially important in inspections).
- Checklists or roles should be used, where appropriate, to increase effectiveness of defect identification; for example, in an inspection, roles such as data entry clerk or technical architect may be required to review a particular document.
- Management support is essential for a good review process (e.g. by incorporating adequate time for review activities in project schedules).
- There should be an emphasis on learning and process improvement.

Other more quantitative approaches to success measurement could also be used:

- How many defects found.
- Time taken to review/inspect.
- Percentage of project budget used/saved.

In his original paper on the benefits of inspections in 1976, Michael Fagan of IBM, who developed the Fagan Inspection Method, reported a 23 per cent increase in 'coding productivity alone' by using inspections. Success can be measured in many ways; however, the key is to keep measuring to ensure success is still being achieved and, more importantly, reported to a wider audience.

CHECK OF UNDERSTANDING

(1) Compare the differences between a walkthrough and an inspection.

(2) Name three characteristics of a walkthrough.

(3) Identify at least five success factors for a review.

STATIC ANALYSIS BY TOOLS

Like reviews, static analysis looks for defects without executing the code. However, unlike reviews static analysis is carried out once the code has been written. Its objective is to find defects in software source code and software models.

Source code is any series of statements written in some human-readable computer programming language that can then be converted to equivalent computer executable code – it is normally generated by the developer.

A software model is an image of the final solution developed using techniques such as Unified Modeling Language (UML); it is normally generated by a software designer.

Static analysis can find defects that are hard to find during test execution by analysing the program code, e.g. instructions to the computer can be in the form of control flow graphs (how control passes between modules) and data flows (ensuring data is identified and correctly used).

The value of static analysis is:

- Early detection of defects prior to test execution. As with reviews, the earlier the defect is found, the cheaper and easier it is to fix.
- Early warning about suspicious aspects of the code or design, by the calculation of metrics, such as a high-complexity measure. If code is too complex it can be more prone to error or less dependent on the focus given

to the code by developers. If they understand that the code has to be complex then they are more likely to check and double check that it is correct; however, if it is unexpectedly complex there is a higher chance that there will be a defect in it.

- Identification of defects not easily found by dynamic testing, such as development standard breaches as well as detecting dependencies and inconsistencies in software models, such as links or interfaces that were either incorrect or unknown before static analysis was carried out.
- Improved maintainability of code and design. By carrying out static analysis, defects will be removed that would otherwise have increased the amount of maintenance required after 'go live'. It can also recognise complex code that if corrected will make the code more understandable and therefore easier to maintain.
- Prevention of defects. By identifying the defect early in the life cycle it is a lot easier to identify why it was there in the first place (root cause analysis) than during test execution, thus providing information on possible process improvement that could be made to prevent the same defect appearing again.

Typical defects discovered by static analysis tools include:

- Referencing a variable with an undefined value, e.g. using a variable as part of a calculation before the variable has been given a value.
- Inconsistent interface between modules and components, e.g. module X requests three values from module Y, which has only two outputs.
- Variables that are never used. This is not strictly an error, but if a programmer declares a variable in a program and does not use it, there is a chance that some intended part of the program has inadvertently been omitted,
- Unreachable (dead) code. This means lines of code that cannot be executed because the logic of the program does not provide any path in which that code is included.
- Programming standards violations, e.g. if the standard is to add comments only at the end of the piece of code, but there are notes throughout the code, this would be a violation of standards.
- Security vulnerabilities, e.g. password structures that are not secure.
- Syntax violations of code and software models, e.g. incorrect use of the programming or modelling language.

Static analysis tools add the greatest value when used during component and integration testing. This will normally involve their use by developers to check against predefined rules or development standards, and by designers during software modelling.

A static analysis tool runs automatically and reports all defects it identifies, some of which may be insignificant and require little or no work to correct, whilst others could be critical and need urgent correction. These defects therefore

require strong management to ensure that the full benefit is obtained from using the tool in the first place.

Software compilers are computer programs (or a set of programs) that translate codes written in one computer language (the source language) into another computer language (the target language). As part of the compile process certain static analysis can be undertaken that will identify some defects and provide the calculation of software metrics.

Static analysis tools are explained in Chapter 6.

CHECK OF UNDERSTANDING

(1) What is static analysis looking for?

(2) Name the five benefits of static analysis.

(3) What is the name of the type of defect that relates to password checks?

SUMMARY

In this chapter we have looked at how review techniques and static analysis fit within the test process defined in Chapter 2. We have understood that a review is a static test, i.e. it is a test carried out without executing any code (by reading and commenting on any document (work-product) such as a requirements specification, a piece of code or a test plan/test case). We have also looked at the different types of review techniques available, such as walkthroughs and inspections, as well as spending time understanding the benefits of reviews themselves.

Reviews vary in formality. The formality governs the amount of structure and documentation that surround the review itself.

To obtain the most benefit from reviews, they should be carried out as early in the project life cycle as possible, preferably as soon as the document to be reviewed has been written and definitely, in the case of work-products such as requirement specifications and designs, before any code is written or executed. The roles of the participant reviewers need to be defined and, in the more structured review techniques, written output from reviews is expected.

We have learnt that static analysis is checking the developed software code before it is executed, checking for defects such as unreachable (dead code) and the misuse of development standards. We have also learnt that static analysis is best carried out using tools, which are described in detail in Chapter 6.

Like reviews, the best benefits of static analysis are realised when it is carried out as soon as possible after the code is written.

Example examination questions with answers

E1. K1 question
Which of the following is most likely to be a benefit of using static techniques?

a. Fewer performance defects.
b. Productivity improvements in the development process.
c. More efficient regression testing.
d. Quick return on investment in static analysis tools.

E2. K2 question
Which of the following has the typical formal review activities in the correct sequence?

a. Kick-off, review meeting, planning, follow-up.
b. Kick-off, planning, review meeting, re-work.
c. Planning, kick-off, individual preparation, review meeting.
d. Planning, individual preparation, follow-up, re-work.

E3. K2 question
Which of the following statements are true?

(i) Defects are likely to be found earlier in the development process by using reviews rather than static analysis.
(ii) Walkthroughs require code but static analysis does not require code.
(iii) Informal reviews can be performed on code and specifications.
(iv) Dynamic techniques are generally used before static techniques.
(v) Dynamic techniques can only be used after code is ready to be executed.

a. (i), (ii), (vi).
b. (ii), (iii), (v).
c. (i), (iv), (v).
d. (i), (iii), (v).

E4. K1 question
Which of the following are static techniques?

a. Walkthrough.
b. State transition testing.
c. Decision table testing.
d. Statement testing.

E5. K1 question
Which one of the following roles is typically used in a review?

a. Champion.
b. Author.
c. Project sponsor.
d. Custodian.

E6. K2 question

Which of the following is most likely to be performed by developers?

a. Technical review of a functional specification.
b. Walkthrough of a requirements document.
c. Informal review of a program specification.
d. Static analysis of a software model.

Answers to questions in the chapter

SA1. The correct answer is a.
SA2. The correct answer is c.
SA3. The correct answer is d.

Answers to example questions

E1. The correct answer is b.
Although the other options might be seen as benefits they are not amongst the most likely benefits. Option (b) is the benefit that is most likely to be realised.

E2. The correct answer is c.
The correct sequence is: planning, kick-off, individual preparation, review meeting, re-work, follow-up. All of the other options have either the activities in the wrong order or activities missing from the strict flow.

E3. The correct answer is d.
The other answers are incorrect because:

ii. Walkthroughs do not require code and static analysis does require code.
iv. Static techniques do not execute the code and therefore can be run before and after the code is ready for execution.

E4. The correct answer is a.
Options (b), (c) and (d) are all dynamic test techniques.

E5. The correct answer is b.
The Author is the only role that is typically used in a review. A Champion might sponsor the review process but is not a defined role within an actual review; a Project Sponsor, if technically competent, might be asked to play a defined role within the review process, but whilst using that role they will not be a Project Sponsor; finally, a Custodian might ensure the results are stored safely but would not be involved in the actual review itself.

E6. The correct answer is d.
Static analysis is done almost exclusively by developers. The other review types would be performed using a combination of developers, testers and other interested stakeholders.